Particle Tracking and Geometry in the MPD.

Near Detector Software workshop

Eldwan Brianne, Tom Junk, Leo Bellantoni,
Thomas Campbell, Gavin Davies
Fermilab, 24th July 2019

DEEP UNDERGROUND NEUTRINO EXPERIMENT

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

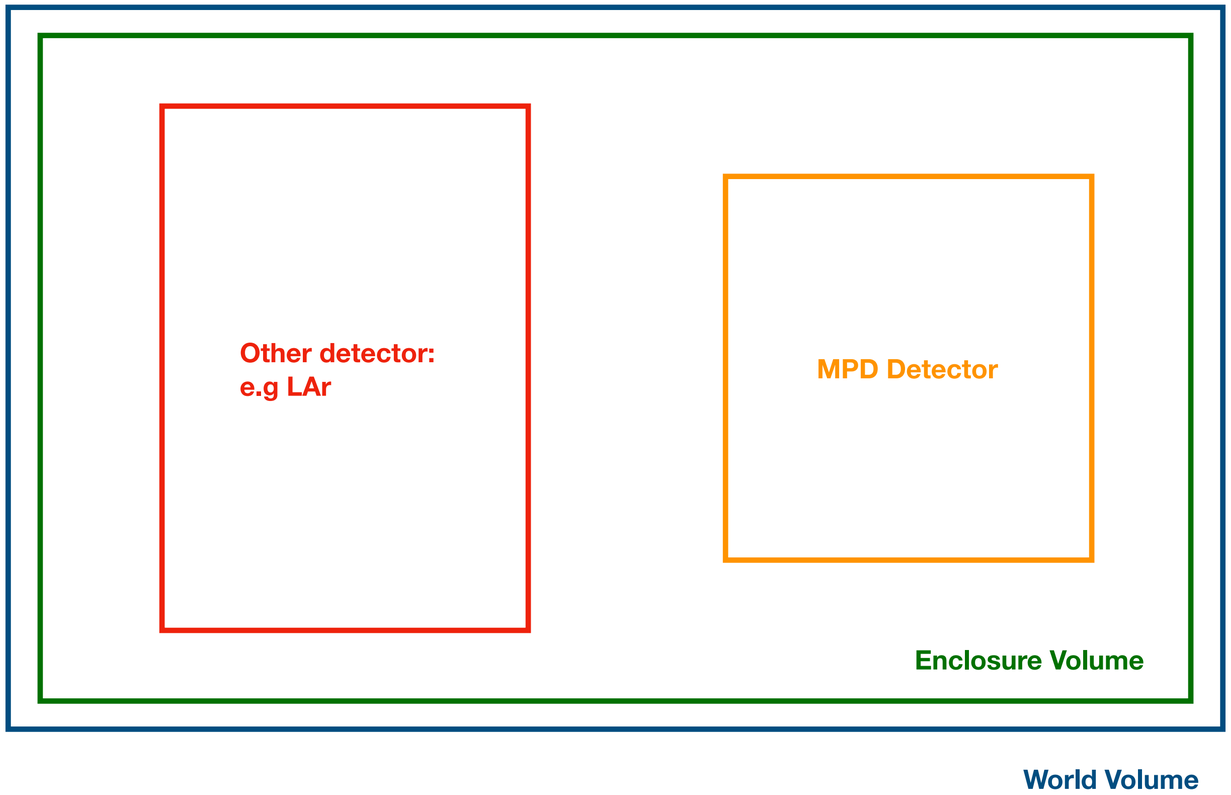
Max-Planck-Institut für Physik (Werner-Heisenberg-Institut)

DESY.

Current MPD Geometry.

The core component

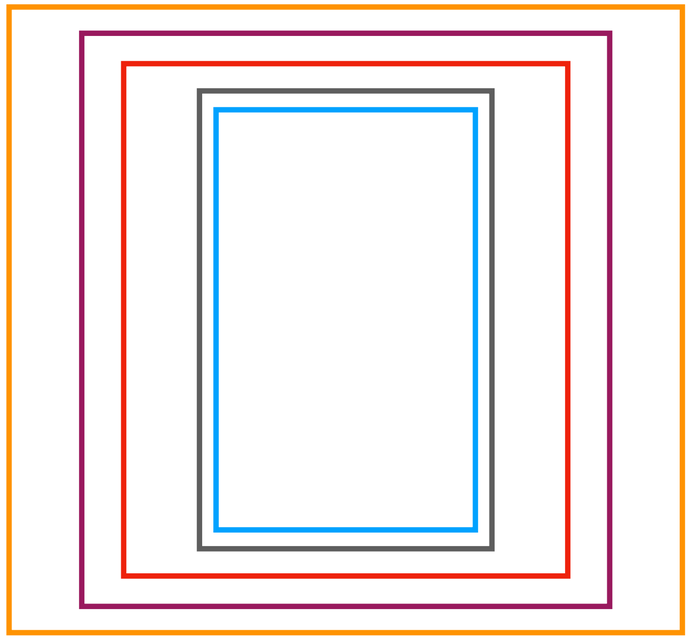
- Geometry generated with gegede and dunendggd
  - Subdetector/NDHPgTPC_v03.py
- Global geometry structure
  - Construct all sub-components: TPC, PV, ECAL Barrel, ECAL Endcap, Magnet
  - Place them in the detector enclosure volume ➠ can place several ND components: ArgonCube, MPD and 3DST
  - Place the enclosure in the world volume ➠ Cavern made of rock (needed for background)
- For development
  - Perform first bullet then just place the detector into a world volume filled with Air.
- Important
  - Each volume has its own coordinates!
  - Enclosure defines position of the sub-detectors and world defines position of enclosure



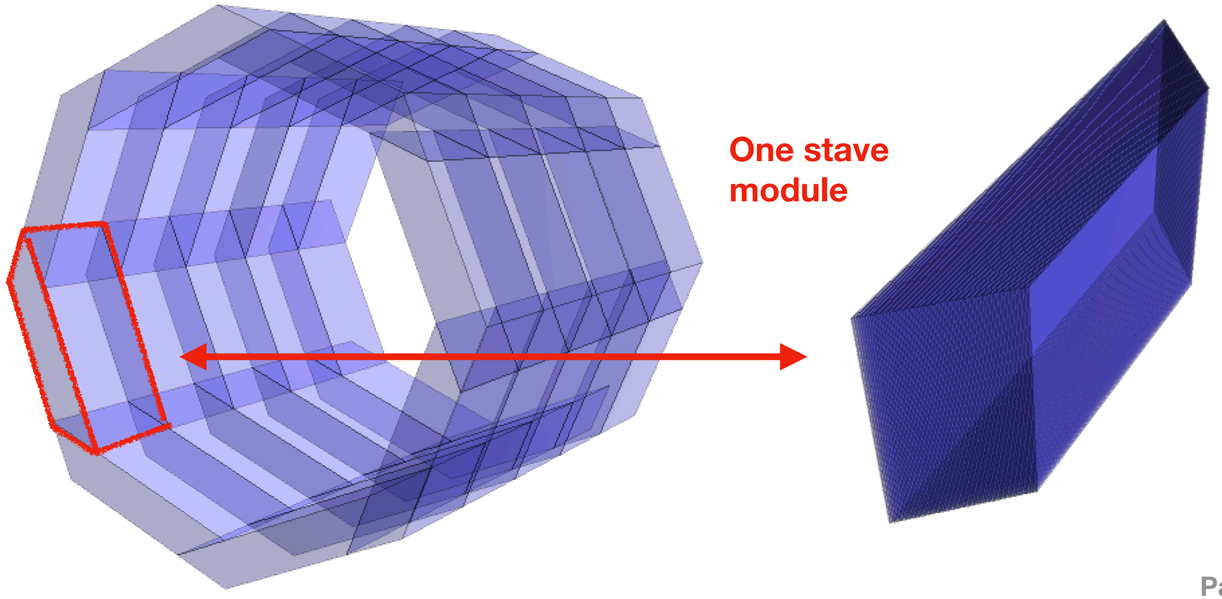
Current MPD Geometry.

Details for the MPD

- Each sub-component are built individually
- For the ECAL Barrel
  - A global envelope is built first (octagonal barrel - Polyhedra Regular)
  - For each octant, the stave is built - this is done to assign different names to staves (later used for segmentation purposes)
  - Each stave can be cut in smaller small modules along the x-axis
  - Each type of layer (pre-built) is then placed accordingl in the stave volume
  - The stave is then placed correctly into the envelope



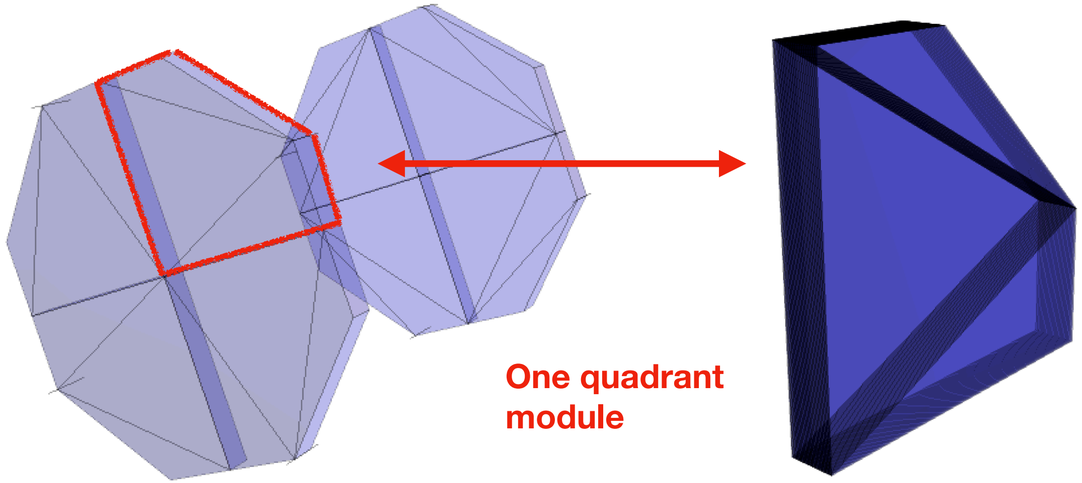
MPD Detector

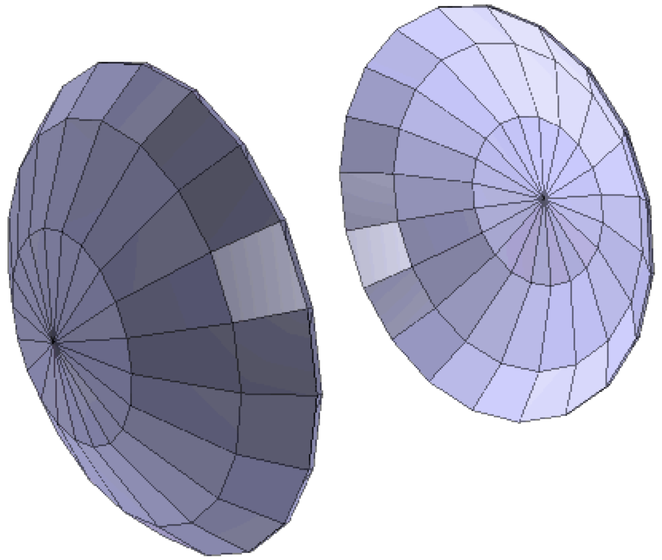


Current MPD Geometry.

Details for the MPD

- For the ECAL endcap
  - Similar ➠ Envelope is subtraction of two Polyhedra Regular
  - Each quadrant is the intersection of a Square and a Polyhedra Regular
  - Each quadrant can be divided in sub-modules ➠ to be done
- For the pressure vessel
  - Similar thing ➠ endcap is intersection of cylinder and sphere
  - Something to improve her) ➠ too much space between PV endcap and ECAL endcap

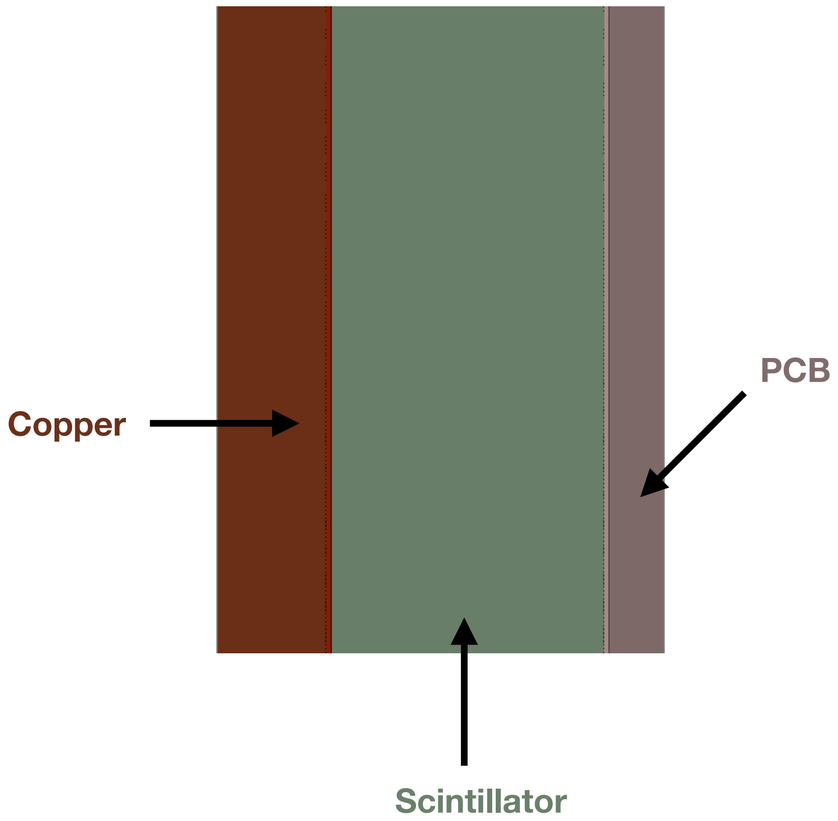




Current MPD Geometry.

Details for the MPD

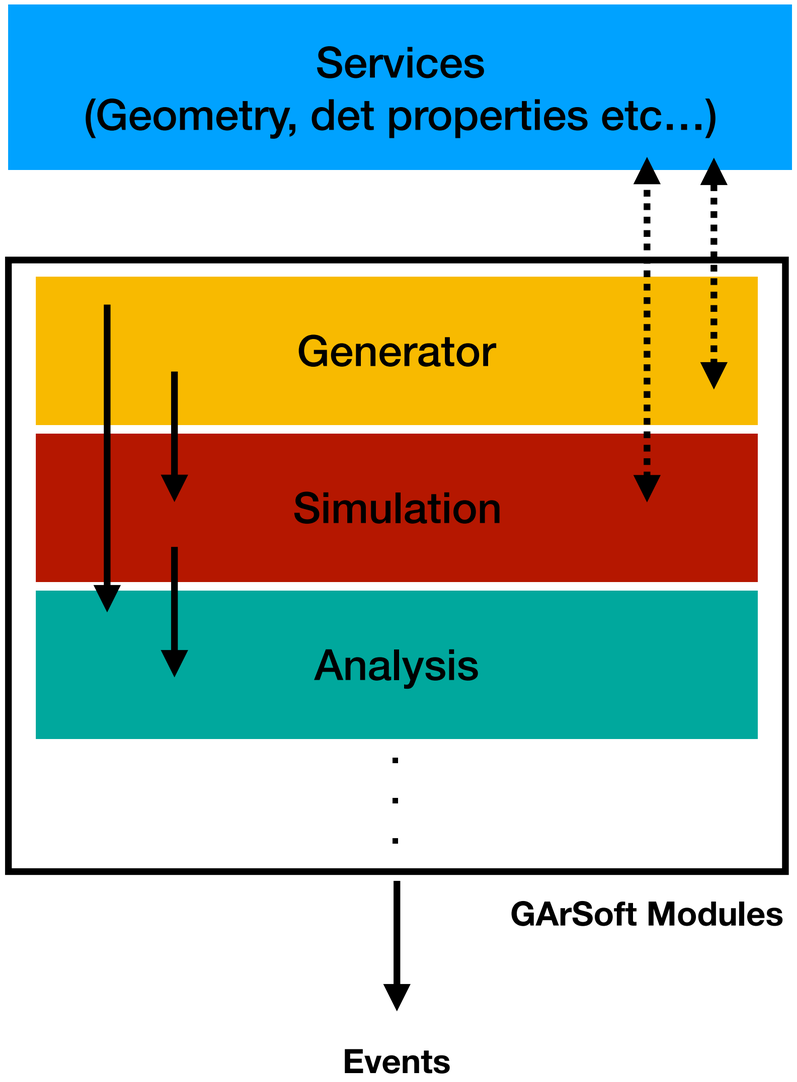
- Layer details
  - Simple model of a layer inserted into the stave
    - Similar to previously - envelope defines the layer volume, insert slices inside of material
  - No description of services/dead zones! ➠ once the design is fixed
  - No description of each individual tile or strip
    - Too much memory and file size would explode ➠ simple scintillator plane.
    - Segmentation in the digitisation phase
  - Several models of layers can be done and then inserted according to the config file (e.g High/Low granular layers have different structure)
- Current
  - 2 mm Cu / 5 mm Sc / 1 mm PCB (FR4)



MPD Simulation.

Handling of Geometry and running the G4 simulation

- MPD Software based on LArSoft ➠ GArSoft
- Configured using fcl files
- Different services to handle different parts
  - Geometry
  - Detector properties
  - …etc…
- Simulation based on LArSoft model
  - Load services ➠ load gdml, channel mapping, segmentation…
  - Call Generator module ➠ generate single particle, GENIE etc…
  - Call GArG4 module ➠ run the simulation



MPD Simulation.

Handling of Geometry and running the G4 simulation

- GArG4
  - Begin job:
    - User limits: step size, regions, range cut
    - Material properties: i.e optical properties
    - Geometry: load gdml in G4
    - Physics list: pre-built or can be custom via fcl
    - User actions: filter particle to keep, energy deposits actions (user defined)
  - Produce:
    - Take generator particles pass them through G4 (G4Helper)
    - Get back particle list, energy deposits and put them into the event

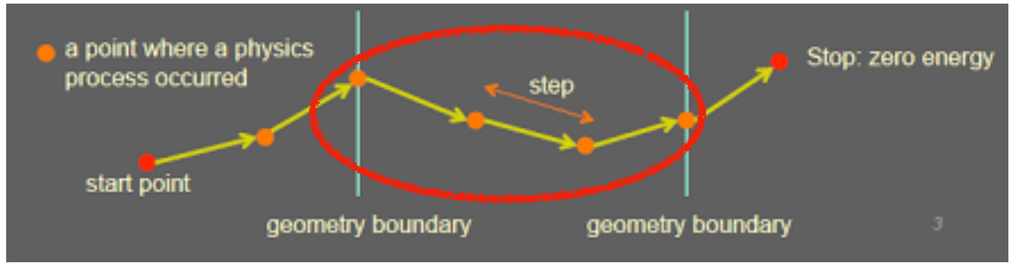
```
// Get the logical volume store and assign material properties
garg4::MaterialPropertyLoader* MPL = new garg4::MaterialPropertyLoader();
MPL->GetPropertiesFromServices();
MPL->UpdateGeometry(store);

this->SetLimitsAndCuts();

// Intialize G4 physics and primary generator action
fG4Help->InitPhysics();
```

```
// add UserAction for handling steps in gaseous argon
fEDepAction = new garg4::EnergyDepositAction(&(*fEngine),
                                             fEDepActionPSet);
uaManager->AddAndAdoptAction(fEDepAction);
```

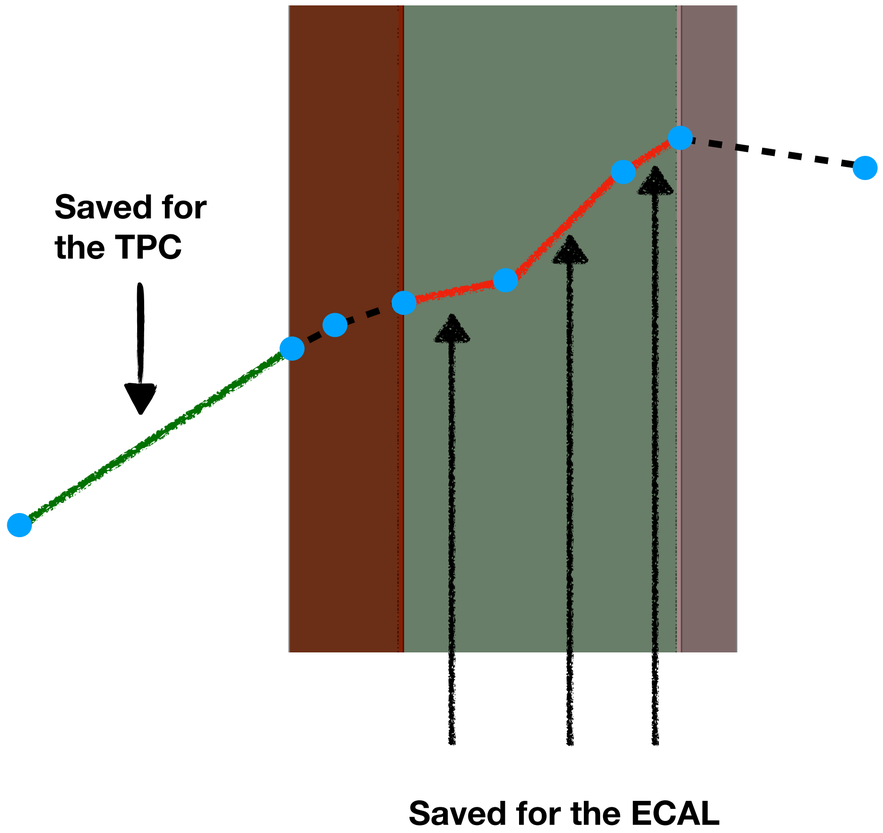
```
  // Now for the sdp::EnergyDepositions
  for(auto const& tpc : fEDepAction->EnergyDeposits()){
    LOG_DEBUG("GArG4")
  << "adding TPC deposits for track id: "
  << tpc.TrackID();

    TPCCol->emplace_back(tpc);
  }

  // And finally the sdp::CaloDepositions
  for(auto const& hit : fAuxDetAction->CaloDeposits())
{
  LOG_DEBUG("GArG4")
    << "adding calo deposits for track id: "
    << hit.TrackID();

  ECALCol->emplace_back(hit);
}
```

MPD Simulation.

Particle Tracking in the MPD



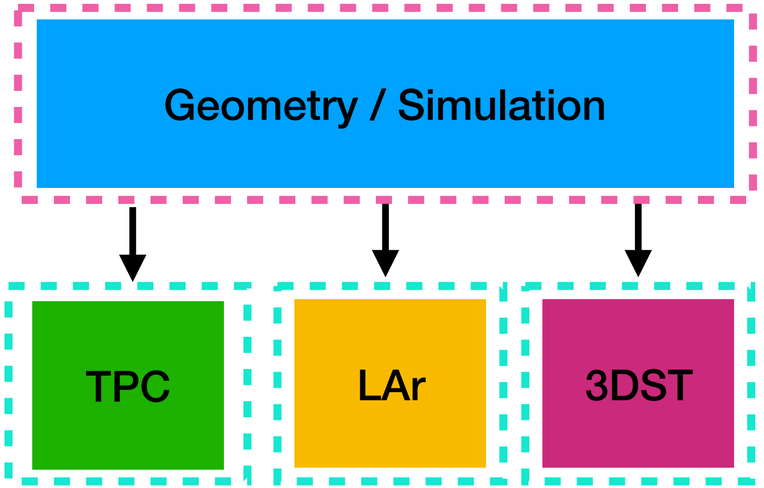
- Tracking is handled by G4
- User specific tracking ➠ User Actions!
  - Used to create simulated hits
- Each MPD detector has its own user action function
  - TPC ➠ EnergyDepositAction
  - ECAL ➠ AuxDetAction
- User Actions can be configured with fcl parameters
- In each class, the Stepping Action is defined
  - Check the position (middle of the step), energy of the step
  - Check material in which the step is ➠ sensitive?
  - If sensitive save the step (energy, position, time, length, id…) into a vector
- Perform this for each particle until below a certain E_{kin}!
- Very modular! But can be long depending on the number of particles to track!

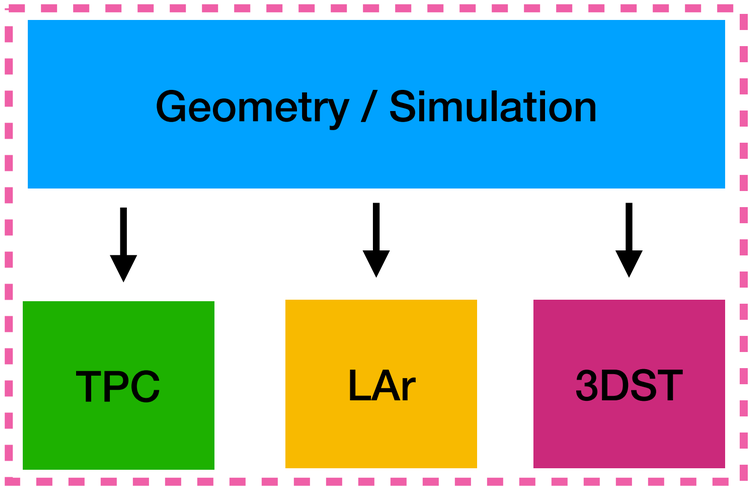


Integration into a common framework.

Ideas

- Need for a common framework in the ND simulation ➠ Not possible to separate the simulations…
- Allows for detector interplay, optimisation…
- What are the needs in term of simulation?
- Possibilities:
  - Create a package specific for the ND simulation containing the common geometry, generators and simulation running code
    - Advantage: Independent of the framework after the simulation
  - Move all sub-detectors to the same framework
    - A lot of work for each sub-detector group depending on the framework chosen…





Conclusion.

Towards a common goal

- The geometry model for the MPD is well advanced and started to be under optimisation
- The simulation framework for the MPD is based on LArSoft, all integrated into a common framework GArSoft
- Going towards a common ND framework
  - Will allow interplay between detectors (i.e LAr/GAr matching…) and also optimisation
  - Common physics analyses?
  - Not so easy depending on the different sub-detector needs
  - Options:
    - Separate simulation/geometry and digitisation/reconstruction ➠ might be the easiest
    - Stick to one framework for all sub-detectors ➠ might be quite some work
- To do list: https://docs.google.com/spreadsheets/d/1DhdW7R8iKR6Aar7AmC-4Lswt-z_RvrImx1bjAPQYa-A/edit?ts=5c58b34b#gid=1386834576
  - Simulation -> integrate other detectors?
  - Improve current and integrate particle reconstruction/identification algorithms (pi0 …)
  - Improve calorimeter clustering
  - Add few physics benchmarks into the analysis
  - Develop calibration method